COMBINATORIAL PROBLEMS OF ENUMERATION IN PROGRAMMING CONTESTS

M. Kinder, M. Falileeva, L. Shakirova

Kazan Federal University (RUSSIAN FEDERATION)

Abstract

This paper describes a recursive approach to the enumeration of some classes of combinatorial tasks. Most tasks are used in the specific scope of teaching and learning informatics through olympiads and other competitions.

Combinatorial problems can often lead to interesting and beautiful dynamic programming tasks, because they both depend on recurrence relations: formulae that solve a larger problem in terms of one or more smaller problems. Combinatorics of course is not the only branch of mathematics that can yield interesting tasks for programming contests. We focus on it here because many combinatorial problems have entertaining legends and they are easily accessible to students.

Many of the examples in this paper are taken from the Open Cup named after E.V. Pankratiev (Grand-Prix of Tatarstan). Full texts for all of these problems are available on the Internet: www.icl.ru/turnir. We hope that some classes of such tasks would enlarge scope of tasks for use in programming contests at various levels.

Keywords: programming contests, informatics olympiads, combinatorial tasks, training.

1 INTRODUCTION

Competitions in informatics were introduced about forty years ago, with the clever ideas of attracting talented young people to the computer science and to the programming. These competitions perfectly demonstrate their efficiency in search of young talents and the formation of high quality specialists in the field of computer technology.

Usually, these programming competitions require students to submit programs which are then run through a variety of test scenarios and judged accordingly. The difficulty, however, lays not so much in the programming but rather the design of the underlying algorithms [1]. In most cases, these contests are based on automatic rating of the submitted solutions. This is accomplished by running them on batches of input data and testing correctness of the output.

The first programming contest hosted in Kazan, the capital of Tatarstan (Russia), in 2000. The 16th international programming contest will be hold in Kazan in 2016. The conditions of contests are as close as possible to the ACM ICPC conditions: the participating team, consisting of three people, has only one computer and five hours to solve 10 to 12 fairly difficult problems. The successful solution of problems requires actual knowledge of programming languages, mathematical training, knowledge of algorithms and data structures, and skill to work in team.

Tatarstan students are often become winners of the final stage of the All-Russian Olympiad for School Students in Informatics. In 2015, the students of Kazan Federal University are reached the final of the prestigious World Championship in Collegiate Programming Contest (Association for Computing Machinery International Collegiate Programming Contest). This competition consists of several stages. In order to reach the final, it is necessary to show the best results in the regional and subregional stages of the competition, which, this year, involved about 13 thousand teams from almost three thousand universities in 90 countries [2].

Every year, teams for sports programming attend summer and winter camp, organized specially for them in Petrozavodsk and Izhevsk, where, besides Russian programmers, teams from Poland, the Czech Republic, Japan, Ukraine and Belarus gather together. In addition, a dozen of open championships are held during the year, where it is possible to meet potential ACM ICPC finalists. Open All-Siberian Programming Contest, Open Championship of the Republic of Tatarstan – Tournament of ICL (International Computers Limited) [3] and Ural Championship are among these events.

2 CONTEST PROBLEMS OF TOURNAMENT OF ICL

Ideas for tasks can come from many different disciplines; however some branches of mathematics are particularly useful for creating informatics olympiad problems. In this section we will focus on tasks derived from the mathematical field of combinatorics.

Combinatorics problems can often lead to interesting or unusual dynamic programming tasks, because they both share a core dependency on recurrence relations: formulae that solve a larger problem in terms of one or more smaller problems [4]. For instance, consider the following problem.

SOLID STACKS [Open Cup named after E.V. Pankratiev, 2014, author – M. Kinder.]

Consider a stack of k boxes. The weight of each box is a positive integer, and the sum of all those weights is n. Additionally, we call stack solid if the weight of each box in the stack is not less than the total weight of all boxes above it. For example, for n = 7, there are two solid stacks of k = 3 boxes: 7 = 1 + 2 + 4 *and* 7 = 1 + 1 + 5*, where the first number denotes the weight of the upper box, the second one is the weight of the middle box, and the last one denotes the weight of the box which is placed on the ground. How many different solid stacks can be built for given n and k?* ($1 \le n \le 10^6$, $1 \le k \le 20$). (Output the number of different solid stacks modulo ($10^9 + 9$).)

To make this more precise, let us say that a partition of a natural number n into k parts is *non-squashing* [5] if when the parts are arranged in nondecreasing order, say

$$n = p_1 + p_2 + \ldots + p_k$$

with $1 \le p_1 \le p_2 \le \ldots \le p_k$, we have $p_1 + p_2 + \ldots + p_i \le p_{i+1}$ for $1 \le i \le k - 1$. If the boxes in a stack have the weights (from the top) $p_1, p_2, \ldots, p_k$, the stack will solid if and only if the partition of a number n is non-squashing. Let $m[n,k]$ be the number of non-squashing partitions of n into exactly k parts. The numbers $m[n,k]$ satisfy the recurrence

$$m[2n, k] = m[2n - 1, k] + m[n, k - 1],$$

$$m[2n + 1, k] = m[2n, k]$$

with initial conditions $m[n, 0] = 0$, $m[n, 1] = 1$ and $m[n, k] = 0$ for all $k > n$. Realization of this algorithm is already not difficult.

The online encyclopedia of integer sequences [6] is full of interesting combinatorial sequences and recurrence relations; simply browsing through the encyclopedia can yield interesting results.

DISPERSED PARENTHESES [Open Cup named after E.V. Pankratiev, 2015, author – M. Kinder.]

The sequence of calculations in arithmetic expressions is usually set by a certain arrangement of parentheses. For example, $(3 \cdot (2 + 1)) \cdot (4 - 5)$. *After deleting all the elements from the expression except parentheses remaining symbols form a* parentheses sequence (())(). *Let's assume that adding character* "0" *does not corrupt the sequence. Let's call such sequence a* disperse parentheses sequence*. Also this can be defined as follows:*

- *An empty line is a disperse parentheses sequence.*
- *If S and T – disperse parentheses sequences, then lines 0S, S0, (S) and ST are also disperse parentheses sequences.*

The depth of disperse parentheses sequence is the maximum difference between the number of opening and closing parentheses in the sequence prefix. (The prefix of line S is the line, which can be obtained from S by deleting symbols from the tail of the line. For example, the prefixes of line "ABCAB" are lines "", "A", "AB", "ABC", "ABCA" and "ABCAB".) Thus, the depth of the sequence "(0)(0())0" equals two (prefix "(0)(0(" contains three opening and one closing parentheses). Calculate the number of possible disperse parentheses sequences n symbols long, that have depth k ($1 \le n \le 300$, $0 \le k \le n$).

(Output the number of possible disperse parentheses sequences n symbols long, that have a depth k mod ($10^9 + 9$).)

Let m[n, k] be the number of possible disperse parentheses sequences n symbols long, that have a depth not more k. For example, m[3, 1] = 4, since there are four disperse parentheses sequences

"0()", "(0)", "()0" and "000". In order to get the answer, you need to evaluate an expression m[n, k] – m[n, k – 1]. Using addition and multiplicative principles effectively can provide insight in solving counting problem. The numbers m[n,k] satisfy the recurrence

$$m[n,k] = m[n-1,k] + \sum_{i=0}^{n-2} m[i,k-1] \cdot m[n-i-2,k],$$

$m[i, 0] = m[0, j] = 1$ for all $i \geq 0$, $j \geq 0$. In general, $m[n, k] – m[n, k – 1]$ is the number of Motzkin paths of length n and height k (see [6], A097862).

MICROCIRCUITS [Open Cup named after E.V. Pankratiev, 2015, author – M. Kinder.]

You probably know how microcircuits look like. First of all it is important to pay special attention to connections. Contacts on the circuit are connected by lines. If two lines do not intersect then there is no connection between respective contacts. You are holding a circular circuit with n contacts around the borderline. You have to calculate the number of possibilities to put exactly k non-intersecting lines each connecting two contacts ($1 \leq k \leq n \leq 40$).

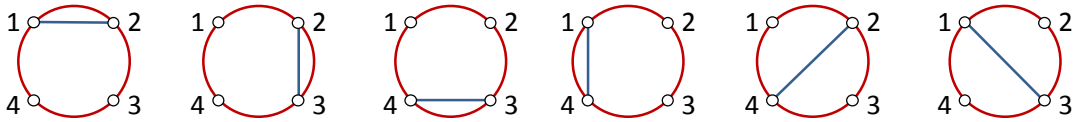

Fig. 1

Let $m[n,k]$ be the number of ways of drawing k non-intersecting chords between n points on a circle; in Fig. 1 it can be seen that $m[4,1] = 6$. In general, $m[n,k]$ is the famous sequence of Motzkin numbers; see [7] for some of the many, many other things that it counts. (For example, $m[n,k]$ is the number of Motzkin paths of length n with k up steps [8].) Using addition and multiplicative principles effectively can provide insight in solving counting problem. In general, a similar argument shows that

$$m[n,k] = m[n-1,k] + \sum_{i=0}^{n-2} \sum_{j=0}^{k-1} m[i,j] \cdot m[n-i-2,k-j-1],$$

$m[i, 0] = 1$ and $m[0, j] = 0$ for all $i \geq 0$, $j \geq 1$. This mathematical recurrence is easy to turn into a dynamic programming relation.

PHARMACIES [Open Cup named after E.V. Pankratiev, 2012, author – M. Kinder.]

Absolutely healthy people live in the city of N. There is only one citizen who is constantly unhappy about this fact. That is the reason he works as a pharmacist and has to sit idly the whole time. Trying to entertain himself, the pharmacist makes up mathematical problems using pharmacy weights and then solves them himself. But one problem turned out to be extremely difficult.

The pharmacist uses a set of n weights of 1, 2^1, 2^2, … , 2^n *grams and a two-pan weighing scale. Weights can be put on both scale pans. Two ways of weighing that differ only in arrangement of the scale pans are considered to be the same. For example, there are two ways of weighing a* 5 *gram weight using three* 1, 2^1 *and* 2^2 *gram weights*: 5 = 1 + 4 *and* 5 + 1 = 2 + 4.

The task of the pharmacist is to find out the number of ways to weigh of m grams using different sets of weights ($1 \leq n \leq 62$, $1 \leq m \leq 5 \cdot 10^{18}$).

This problem is closely related to the discussed above idea of recurrence relations and of dynamic programming. The full analysis of the solution is also available on web site of Tournament ICL [3].

3 CONCLUSION

Creating high quality tasks is a difficult and time-consuming process. We endeavour to make tasks interesting, understandable and accessible. This paper describes a recursive approach to the enumeration of some classes of combinatorial tasks. Combinatorial problems can often lead to interesting or unusual dynamic programming tasks. Combinatorics of course is not the only branch of

mathematics that can yield interesting tasks for programming contests. We focus on it here because many combinatorial problems have entertaining legends and they are easily accessible to students.

Many of the examples in this paper are taken from the Open Cup named after E.V. Pankratiev (Grand-Prix of Tatarstan). Full texts for all of these problems are available on the Internet [3]. We hope that some classes of such tasks would enlarge scope of tasks for use in programming contests at various levels.

REFERENCES

[1] Burton, B. (2008). Informatics olympiads: challenges in programming and algorithm design. In: Dobbie, G. and Mans, B. (Eds.) Proceedings of Thirty-First Australasian Computer Science Conference (ACSC 2008), Wollongong, NSW, Australia. CRPIT, 74, 9–13.

[2] Maiatin, A., Mavrin, P., Parfenov, V., Pavlova, O., Zubok, D. (2015). The Estimation of Winners' Number of the Olympiads' Final Stage. Olympiads in Informatics, vol. 9, pp. 139–145. DOI: http://dx.doi.org/10.15388/ioi.2015.11.

[3] www.icl.ru/turnir

[4] Burton, B., Hiron, M. (2008). Creating informatics olympiad tasks: exploring the black art. Olympiads in Informatics, vol. 9, pp. 16–36.

[5] Sloane, N., Sellers, J. (2005) On non-squashing partitions. Discrete Mathematics, 294, vol. 9, pp. 259–274.

[6] http://oeis.org

[7] http://oeis.org/A080159

[8] http://oeis.org/A055151